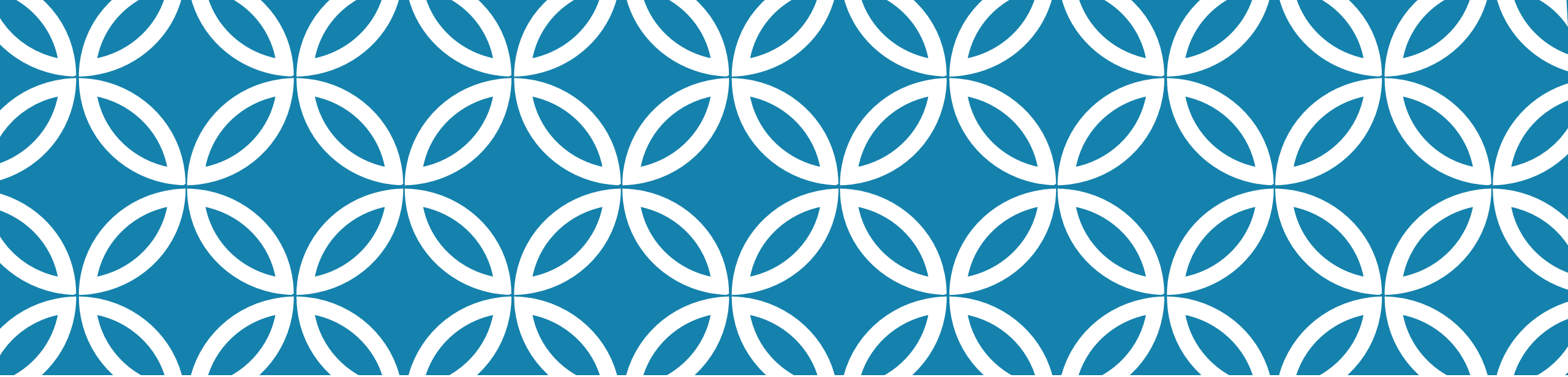

MULTIPARADIGM LANGUAGES

OPTIONAL COURSE

SEMESTER 6

introduction

course organization

WHAT IS A PROGRAMMING PARADIGM?

"A way

of conceptualizing what it means to perform computation,

of structuring and organizing how tasks are to be carried out on a computer"

[Timothy A. Budd. Multiparadigm Programming in Leda, Addison-Wesley, 1995.]

"A collection of concepts that together guide the design process and determine the form of valid programs."

[A.L. Ambler, M.M. Blumett and B.A. Zimmerman. Operational versus definitional: a perspective on programming paradigms. Computer Journal. 1992, pp. 28-42]

PROGRAMMING LANGUAGE

= a notation that is intended for the expression of computer programs and is capable of expressing any computer program [BJ. MacLennan. Principles of Programming Languages, 2nd edition. H.R.W.,1987.]

Purposes of programming languages:

- to express programs so that they can be executable on a computer.
- To describe programs to programmers.

Requirements:

- universal
- natural
- implementable

The fifth generation (1980++) of programming languages =>
reflects patterns of thoughts

WHICH COMES FIRST? LANGUAGES OR PARADIGMS

Traditionally, languages came first and paradigms were abstractions of features of existing languages.

We can say languages induced paradigms

[Brent Hailpem. Introduction to multiparadigm languages. IEEE Software, 1986. pp. 6-9].

- This is the relationship between languages and paradigms from the *first* to the *fourth* generation of languages

The fifth generation languages are paradigm-induced languages

- programming languages are designed and implemented based on the notions of the underlying paradigms

a language supports a paradigm if the language makes a paradigm ***convenient***

[R. Floyd. The paradigms of programming. Comm. ACM, Vol. 22, No.8, pp. 455-460.1979].

- if a language makes a paradigm *feasible*, but *not convenient*, it weakly supports the paradigm.

TWO PRIMARY PARADIGMS: IMPERATIVE AND DECLARATIVE

- ***imperative:*** *instructions related to changing the state*

examples:

- **procedural**: groups instructions into procedures,
- **object-oriented**: groups instructions with the part of the state they operate on,

- ***declarative:*** *declarations of properties of the desired result (no state!)*

examples:

- **functional:** the result is declared as the value of a series of function applications,
- **logic:** the result is declared as the answer to a question about a system of facts and rules,
- **reactive:** the result is declared with data streams and the propagation of change

PARALLEL/CONCURRENT PARADIGMS

- Implicit vs. Explicit parallelism (levels of abstractions for parallelism)
- Task vs. Data parallelism
- Synchronicity vs. Asynchronicity

- Data-flow
- Actors

=> languages constructions that facilitates concurrency and parallelism

MULTIPARADIGM LANGUAGES ••• NOT A DREAM BUT A NEED

- Programmers can choose the most appropriate paradigm for a particular problem so that the gap between the design phase and the programming phase can be minimized.
- Existing software (e.g. library subroutines) written in languages of a paradigm is reusable in languages of another paradigm.
- It is natural to model real-world objects using multiparadigm languages since they can be viewed as loosely coupled distributed systems with multiparadigm cooperating subsystems.

[Chi-keung Luk. The Design and Implementation of a Multiparadigm Programming Language. PhD thesis. The Chinese University of Hong Kong, 1993]

NEW LANGUAGES ARE MULTIPARADIGM LANGUAGES

augmented languages

- add additional paradigms to an existing language to permit users to utilize a new programming style without learning a completely new language
- through new language construction or through libraries

For example:

- C++ extends C with support for object-oriented programming
- Java 8/ C# – introduce functional programming concepts and constructs

multiparadigms languages

- built to support more than one paradigm – e.g. **Scala, Kotlin, Julia, ...**

COURSE ORGANIZATION

Content:

Programing paradigm overview

Basics of Scala Programming

Object oriented programing in Scala

Functional programming in Scala

Concurrency in Scala

Parallel programming in Scala

Introduction in Scala coroutines

Introduction in Actor programming with Scala (Akka)

Student work:

- **Exercises in Scala** (40%)
  - at the laboratory
- **Essay and project** (60%):
  - choose a multiparadigm language(different from Scala)
  - **essay:** language characteristics
  - **project:** a problem implementation in the chosen language
  - presentation at the end of the semester

QUESTIONS …

Contact:

Virginia Niculescu

virginia.niculescu@ubbcluj.ro